自然言語の意味論 : 概説(ソフトウェア科学・工学の数理的方法)

AUTHOR(S):

郡司, 隆男

CITATION:

郡司, 隆男. 自然言語の意味論 : 概説(ソフトウェア科学・工学の数理的方法). 数理解析研究所講究録 1984, 511: 156-168

ISSUE DATE:

1984-02

URL:

http://hdl.handle.net/2433/98324

数理解析研究所講究録
第511巻 1984年 156-168

自然言語の意味論 (概説)

豊橋技術科学大学 郡司隆男 (Takao Gunji)

1. 構文論、意味論、運用論

自然言語の理論的な研究には様々な分野があるが、中でも次の三つの分野を特に問題にすることがある (Morris 1938)。

(1) a. 構文論 (syntax) ・・・ 記号内部の関係を研究する分野
b. 意味論 (semantics) ・・・ 記号とそれが指すものとの関係を研究する分野
c. 運用論 (pragmatics) ・・・ 記号とそれが指すものとそれを使用するものとの関係を研究する分野

もちろん、他に音韻論とか社会言語学、心理言語学などの分野があるが、上の三つは比較的互いの関連が強いものである。

この三つの区別は重要である。例えば、直観的におかしな文の不都合さの原因もこのどれかで説明されるが、その説明のしかたに違いがある。

(2) a. 私はた食べんさま。
b. 私は馬鹿だことは周知の事実だ。
(3) a. 私は昨日赤い無色の液体を飲みほした。
b. 私は昨日ソフトウェアの信頼性に恐喝された。
(4) a. 私がこちらに向かって歩いてきた。
b. 私はあなたが私があなたが私のことを信じていないということを知っていると思っていることに気付いている。

ここで、(2) は構文論的におかしな文の例であり、(2a)では非語 (「んさま」)、語順の間違い (「た食べ」の中の「た」と「食べ」の順)、句順の間違い (「私は」、「た食べ (=食べた)」、「んさま (=さんま)」の順) 等の誤りがある。(2b)はもう少し微妙な場合であり、いわゆるテニヲハの誤りである。日本語を母国語とする人には (意識しているにせよしていないにせよ) (2b)は正しくは「私が馬鹿なことは周知の事実だ」でなくてはならないことはすぐにわかるであろう。実は、日本語には埋め込み文の中では「は」は使えないという文法規則があるのだが、そういう文法規則を習わなくても普通の日本人は正しく「が」を使うことができる。これは日本人が日本語を母国語として覚える過程で文法を内在的な知識として自然に身につけるからである。

(3) は(2) に比べると文法的な誤りはない。(3) のおかしさはその文を解釈したときに初めて生じるものであり、文法的な誤りとは区別しなければならない。(3a)では「赤い」という語と「無色の」という語が同一の語「液体」を修飾している。一般に、複数の修飾語が一つの名詞にかかっていても文法的な誤りとはならない（例えば「赤い透明なドロッとした血生臭い液体」）。(3a)の問題点は、この場合特定の二つの修飾語――「赤い」と「無色の」――が意味的に矛盾していることにある。この意味で(3a)は意味論的におかしな文の例である。また、特定の動詞が特定の意味範疇の名詞を要求することがある。(3b)の「恐喝する」という動詞はその主語が人間か、そう見なせるものでなくてはならない。従って(3b)は「ソフトウェアの信頼性」を人間に近いものと見なしていることになり、ソフトウェア技術者の悪夢の中ならばいざ知らず、一般には意味をなさない文ということになる。

(4) はさらに、言葉の使用者が関わってくる運用論的なおかしさの例である。(4a)において「私」という言葉は話し手が誰であるかが確定して初めてその意味が決まるような語である。つまり文脈から独立のそれ自体の意味というものはない。こういう表現を指示的表現と呼ぶ。「こちら」、「～てくる」などの表現も指示的表現の一種である。指示的表現のこういう性質を考えると、(4a)は通常の物理的・心理的状況では適切な文脈を考えられないという点でおかしな文ということになる。（もちろん、ここでは分身の術とか分裂した自我とかは考慮の対象外としている。）(4b)はもう少し実際上の問題である。言語処理機械としての人間は恐ろしく高能力であるとともに、また計算機にもできるような単純なこと、例えば暗記するとか、は割合苦手である。(4b)の唯一の問題点は、この観点からいうとその長さである。それもただ長いだけならば大した問題ではないが、(4b)は一つの文の中にどんどん別の文が埋め込まれていくという形で長くなっている。一般にこういう中央埋め込みは解析に余計な負担がかかり、埋め込みが深くなると容易に解析能力の限界を越えてしまう。(4b)は普通の人々には図解でもしてみなくては何度読みかえしてもその意味はとれないだろう。これでも文法的・意味的におかしいところは一つもないのである。

以上で「意味論」と言ったときに大体どういうことが問題とされているかが分ってもらえたと思う。もちろん、「意味」という言葉ほど様々な意味に使われる言葉はなく、言語学者の中でも必ずしも唯一の見解があるわけではないが、ここでは「意味論」の守備範囲を一往上でスケッチした程度のものとして話を進めることにする。

2．　言語学における「意味論」

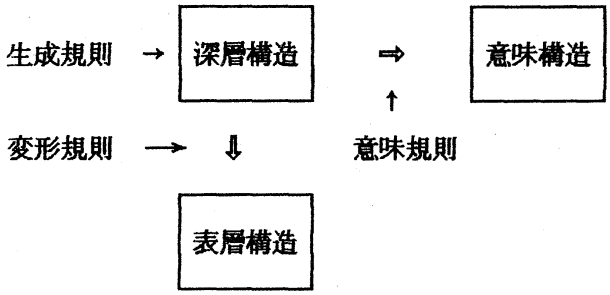
まず、そもそも言語学では一体「意味論」をどのように扱ってきたかをまず振り返ってみたい。言語学と言ってもとてもすべての学派を扱うわけにはいかないし、またその必要もない。ここでは1950年代からアメリカを中心に発展してきた言語学の一つの理論である生成文法を例にとって「意味論」を考えてみたい。生成文法では文法というものをその言語に属するような語の連鎖をすべて、そしてそれのみを生成する規則の集合としてとらえる。こういう考え方は多分ソフトウェア技術者

にとっては目新しいものではないだろう。プログラム言語や、もっと抽象的な形式言語のためのいわゆる形式言語理論はもともと自然言語のための理論である生成文法理論が発展したものなのである。

自然言語のための理論としての生成文法はその初期から独特の構造を持っていた。それは変形という、プログラム言語のための文法にはないものを持っている文法で、そのために変形生成文法、あるいは変形文法と呼ばれている。以下、この変形文法の大まかな成立ちとその変遷を垣間見ることにしたい。

初期の変形文法では表層の、通常読んだり、話したりするままの言葉の構造に加えて、意味の構造により近いものとしてより深い構造を設定した。表層の構造とこの深層の構造とは変形規則で結びつけられる。従って、文法の全体の概念的な構成は次のようになる（Chomsky 1965）。

(5)



上図に示されているように、意味というものは深層構造が入力となって何らかの意味規則の出力として得られる。この意味規則の具体的な形は初期の変形文法ではついに明らかにされなかったのだが、意味そのものの形が分らなくとも意味が同じかどうかというたぐいの議論は可能である。以下、上の図だけからでも言えることを述べてみたい。

まず、意味的な主語、つまり、表層の構造からは隠れている語句の問題がある。これは日本語の場合、使役文などに典型的に現れる。

(6) 表層： a. 私は計算機に日本語を理解させようとした。

深層： b. [私は計算機に [計算機が日本語を理解する] させようとした]

(6a)では「計算機」という言葉は表層では主文の目的語として一度現れるだけで、「理解する」という動詞そのものの主語は明示されていない。これに対して意味規則への入力である深層構造(6b)では、この隠れた主語が補われている。こうしておけば、たとえ将来どのような意味規則ができようとも、表層構造そのままが入力されるよりは扱いやすいだろう。これが初期の考え方であった。(6b)の深層構造から(6a)の表層構造を導くには、同一の名詞句を削除するという変形規則を仮定して用いる。

次は同義性の問題である。自然言語では一つのことを言い表わすのにいくつもの言い方があることがある。例えば受身文と能動文とは細かいニュアンスを別にすれば大体同じ意味を表わす。

(7) 表層： a. 私は計算機に馬鹿にされた。
b. 計算機が私を馬鹿にした。
深層： c. 計算機が私を馬鹿にした

この場合、複数の表層構造が唯一の深層構造から導き出されると考えればよい。上の(7a)と(7b)とに対してただ一つの深層構造、例えば(7c)、を割り当てておけば、意味規則がどのようなものであっても同義性は保証されるだろう。初期にはこのように考えたのであった。この場合、(7a)は受身文を作る変形規則によって導かれる。

次に、同義性と丁度逆の関係である多義性を考えてみよう。次の(8a)の文には少なくとも二つの解釈がある。

(8) 表層： a. 私は計算機に自分の言葉を話させようとした。
深層： b. [私は計算機に [計算機が計算機の言葉を話す] させようとした]
c. [私は計算機に [計算機が私の言葉を話す] させようとした]

上の一つの解釈は「自分」が「計算機」を指す場合である。この解釈をもつ文は(8b)の深層構造から派生されると考えればよい。一方、「自分」が「私」を指す解釈も存在する。この場合には(8c)の深層構造から文が派生されたと考えれば説明できる。つまり、多義性をもつ文には複数の深層構造を割当ててやればよい。表層構造は文の主語と同じ名詞句を「自分」に書き換えるという変形規則により、(8b)または(8c)から導かれる。

このように、限られた範囲の中とはいえ、初期の変形文法（これを「標準理論」という）はそれなりに意味的な現象を扱うことができた。もちろん、意味とは何かという根本の問題には一向に答えてはいないわけだが。

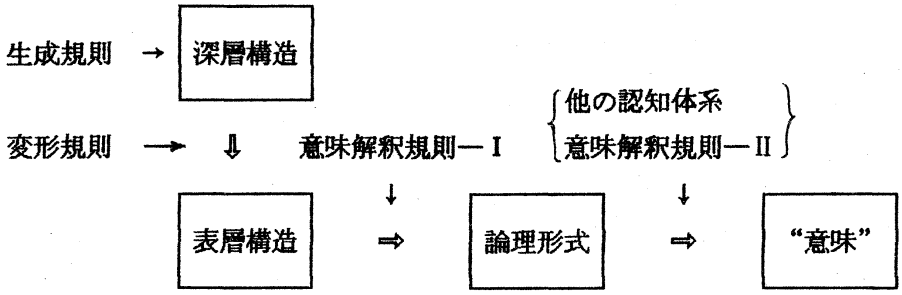
その後の変形文法の研究が進むにつれて、深層構造が一切の意味情報を担っているという仮定には反例のあることがわかってきた。例えば限量詞が絡んでくると、受身文は対応する能動文と同義ではなくなる。

(9) a. 近代人は皆少なくとも二つのプログラム言語を話す。
b. 少なくとも二つのプログラム言語が近代人皆によって話される。

(9a)は一人一人の近代人が別々の二つのプログラム言語を話すという状況で真だが、(9b)は万人に共通な二つの言語（例えば Prolog と Smalltalk）をすべての近代人が話すという状況で真となる。これは深層構造だけを意味規則への入力としていたのでは説明できないことである。

その後いろいろと紆余曲折があったが、最近の「拡大標準理論」と言われる枠組みでは変形規則は表層構造に意味解釈への情報を残し、意味規則は表層構造のみを入力とするというふうに変ってきている。次の(10)に最近の枠組みの概念的な構成を示す。

(10)



この理論の詳細は余りにも専門的になりすぎるので触れないが、表層構造がより抽象化されて意味規則への情報が付加されていることに注意してほしい。例えば先程の(6) の表層構造は概ね次のようになる。

(11) 表層： [私は計算機に [PRO 日本語を理解する] させようとした]

ここで、 PROは音形のない抽象的な代名詞であり、表層構造で「理解する」の形式的な主語となっている。(11)の構造は上の「意味解釈規則－Ｉ」の入力となり、PRO の指すべきものが「計算機」であると決定される。

このような変形文法の意味論はそれなりに、成果を挙げてきてはいるが、次に述べる論理学の立場から見ると長い間不満足なものであった。その最大の弱点は意味構造・論理形式自体が長い間不明確であったことである。つまり、初めに述べた意味論の役目の中の「記号が指すもの」にまで言及した意味論とは程遠いものであった。それでもある程度の説明が可能であることは上で見たとおりだが、このような「意味論」でできることにはやはり限界がある。変形文法における意味論をこの方向で整備しようという動きは70年代の終り頃から現在に至ってようやく見られるようになってきているが、その成果は、多くを今後にまたなくてはならない。

３． 論理学における意味論

一方、これに対して論理学では意味論はどのように扱われてきただろうか。論理学で対象とする「言語」は自然言語ではなく論理という形式言語であるので、それなりに整った意味論を用意することは容易であった。次は論理学における意味論の典型的なものである。

(12) 「雪が白い」は雪が白いときにのみ真だ。

(12)は決して冗談でも無意味な言明でもない。ここで注意しなくてはならないのは「対象言語」と「メタ言語」の区別である。上で「」の中にはいっている部分が対象言語であり、我々が今それについて語っている言語である。ある一つのレベルの言語について語るにはより高いレベルの言語が必要である。これをメタ言語という。(12)では、「」も含めて、「」の中の部分以外がすべてメタ言語となる。対象言語の「雪」という言葉は現実世界に存在する雪を指し、「白い」という言葉は現実世界の白いものの集合を指す。これらの現実世界の中に存在するものの間の関係を記述しているのが「」に続く部分のメタ言語である。上ではたまたまメタ言語として対象言語と同じ日本語を

使ったが、もちろんこれは別々の言語であってよく、またメタ言語が自然言語である必要もない。次はプログラミング言語風の記述である。

(13) val(「雪は白い」) = if val(「雪」) is-in val(「白い」) then true
else false

このような意味論を組織的に与える方法として「モデル理論」がある。ここでいう「モデル」とは大雑把に言って、「指すもの」の集合と対象言語から「指すもの」への写像とを一まとめにしたものである。以下、具体的に一階の述語論理の意味論をモデル理論に基づいて記述してみることにする。

(14) 述語論理の構文論と意味論

対象言語： 変数の集合 Var、定数の集合 Con、述語の集合 $Pred^n$ ($n \geq 1$)、
項の集合 Term = Var ∪ Con

モデル： $M = (D, 【】 g, g)$

値の割当て：

$\alpha \in Var$ ならば $【\alpha】 g = g(\alpha) \in D$

$\alpha \in Con$ ならば $【\alpha】 g \in D$

$\alpha \in Pred^n$ ならば $【\alpha】 g = D^n$ から {0, 1} への函数
(以下、0は偽、1は真と考える。)

	構文論	意味論
a.	$t1, \ldots, tn \in Term, P \in Pred^n \Rightarrow P(t1,\ldots,tn) \in L$	$【P(t1,\ldots,tn)】 g = 【P】 g(【t1】 g, \ldots, 【tn】 g)$
b.	$\alpha, \beta \in L \Rightarrow \alpha \wedge \beta \in L$	$【\alpha \wedge \beta】 g = 1$ iff $【\alpha】 g = 1$ かつ $【\beta】 g = 1$
c.	$\alpha \in L \Rightarrow \sim\alpha \in L$	$【\sim\alpha】 g = 1$ iff $【\alpha】 g = 0$
d.	$\alpha \in L \Rightarrow \forall x\ \alpha \in L$	$【\forall x\ \alpha】 g = 1$ iff すべての $d \in D$ について、$gd(x) = d$ となるような gd に対して $【\alpha】 gd = 1$

ここでの特徴は、意味論が「記号が指すもの」との関係を規則的に示していることである。しかもその規則が構文論と意味論とが全く並行する形で与えられている。例えば、上の(14a) は項と述語から式をつくる構文論の規則と、同時に、そうしてできた式の意味が項の意味と述語の意味から

どのように計算されるかを示す意味論の規則も与えている。これは先に見た変形文法のやり方とは大きく違っている。変形文法ではまず文を生成しておいてからその文を丸ごと意味規則に引き渡しているが、論理学ではいわば、一つ一つの部品の段階でその部品に部分的な意味をくっつけておいて、部品を組み上げるにつれてだんだんと大きな意味を作り上げていくという形をとる。このやり方はFrege によって提唱された「構成的意味論」の考え方である。Frege の原理に従うと、構文論を意味論への考慮をぬきにして組み立てるわけにはいかなくなる。なぜなら、構文的なまとまりがそのまま意味的なまとまりでなくてはならないので、構文的な構造をそのまま意味的な構造としなくてはならないからである。このことは、次に見る自然言語のモデル理論による意味論を考えるともっと明らかになるだろう。

4. Montague の意味論

以上見た論理学の方法を自然言語に適用することは可能だろうか。もしそうならば、自然言語に対しても、「指すもの」にまで言及した意味論を作ることができるだろう。そう考えて、実際にそのような意味論を英語の断片に対して作ってみせたのが論理学者の Richard Montague であった（Montague 1973)。彼の理論は一般に難解だと思われているようだが、実は先程の述語論理の構文論と意味論に比べて本質的に違う点があるわけではない。今日、一階の述語論理程度の内容はソフトウェア技術者にとっても難解ではないだろうから、Montagueの理論を理解するのに大きな困難はないはずである。もっとも、その言語学的内容については、言語学者でも理解するのに時間がかかったわけで含蓄は深いのだが。

まず、最も簡単な例で彼の理論の雰囲気をつかんでみよう。

(15) 計算機は信頼できない。

この文の意味は先程の Fregeの原理によって、その部分、すなわち、「計算機」と「信頼できない」の意味から計算される。ここでこの各々の意味を与える一つのやり方は次のようなものである。

(16) a. 【計算機】＝計算機が持っている性質に対しては１、他に対しては０を与える函数

具体的には例えば、

気が利く	→	0
賢い	→	0
頑固である	→	1
信頼できる	→	0
しばしば壊れる	→	1
物分りが良い	→	0
時々拗ねる	→	1
・・・		

b. 【信頼できない】＝「信頼できる」という性質に対して０を与える函数に対して１を与える函数

ここで、「計算機」の意味として物の集合ではなく、性質の集合が考えられていることに注意してほしい、上の具体例では、1という値を与える性質ばかりを集めた集合、すなわち｛頑固である，いつかは壊れる，時々拗ねる，．．．｝が「計算機」の意味なのである。

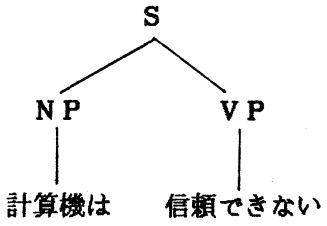
次に、単純な文の構文論と意味論として次の規則を考える。

(17) 構文論: αが名詞句、βが動詞句 ⇒ 「αβ」は文

意味論: 【αβ】＝【β】（【α】）

ここで注意することは、初期の変形文法のように、意味論を与えるために深層構造を設定する必要はないということである。なぜなら、（表層の）構造がそのまま意味構造となっているからである。(17)の規則により、(15)の文は次の（18a)のような構造をもち、その意味は（18b)によって計算されることになる。

(18) a.

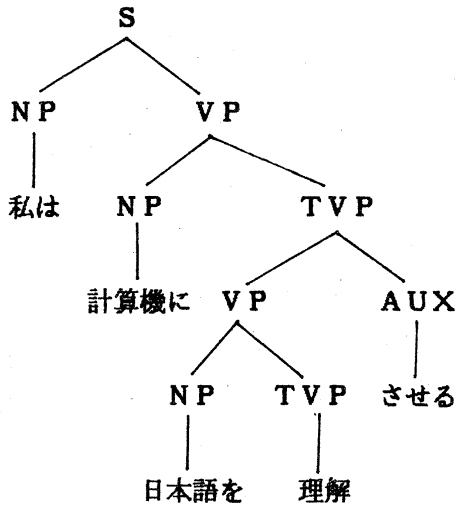


b. 【S】＝【VP】（【NP】）

(18)の構造と意味規則、および、(16)に与えられた個々の語句の意味から、文全体の意味、すなわち【計算機は信頼できない】、は【計算機】が「信頼できる」という性質に対して0を与える函数であるときにのみ、つまり、【計算機】（信頼できる）＝0であるときにのみ、1となることが分かる。上の具体例では、この文の意味はたまたま1（真）である。

次に、もう少し複雑な例を見てみよう。下の(19)は使役文の例である。使役文においては「させる」という助動詞が動詞句を補語としてとっている（Gunji 1983）。

(19) a.



b. 【S】＝【VP】（【NP】）

c. 【VP】＝【TVP】（【NP】）

d. 【TVP】＝【AUX】（【VP】）

「私は計算機に日本語を理解させる」という文の意味は（19a)の構造と、（19b)～(19d）の意味規則によって決定される。以下、順にその過程を追っていってみよう。まず一番基本になる語句「させる」の意味を決めておかなくてはならない。これは現実に日本語としてこの語がどのように使われているかに基づいて決められるが、細部にこだわらなければ概ね次のようになる。（ここでは便宜上メタ言語として数式を含む日本語を使用する。これは前述のように、英語でも、プログラミング語でも、あるいは、Montagueが使用したように論理語でも一向に構わない。）

(20) 【させる】＝VPの意味δの函数Fで、すべてのNPの意味α、βに対して、αが原因となってδ（β）が1になる（すなわち、βがδをする）ときにのみ、F（δ）（β）（α）＝1となるようなもの

ここで函数Fに一つ引数を与えるとその結果が第二の函数となり、さらに第二の引数をとることに注意してほしい。従って、上でF（δ）は第二の函数であり、βという第二の引数をとることができる。この場合、F（δ）（β）はさらに第三の函数となり、第三の引数αをとっている。このことは、すなわち、助動詞（AUX）「させる」の意味は動詞句（VP）の意味を引数として他動詞句（TVP）の意味相当の函数になり、その他動詞句の意味は名詞句（NP）の意味を引数として動詞句の意味相当の函数になり、その動詞句の意味は名詞句の意味を引数として文（S）の意味相当のもの（0か1かの真理値）になるということを表わしている。(19a）の構造から分かるように、この場合、木の上で親の意味は函数としての妹（右側の娘）が引数としての姉（左側の娘）に適用した結果になっている。日本語の意味構造はこのようになっていることが多い。いわば、末っ子が一切を取り仕切るのであり、その最大の末っ子は文末の動詞・助動詞・終助詞である。

さて、(20)の文は次のような段階を経て解釈される。

(21) a. 【日本語を理解（する）】＝「日本語を理解する」という性質に対して1を与える函数に対して1を与える函数（簡単のためにこれ以上分解して考えない）。

b. 【日本語を理解させる】＝NPの意味βの函数Gで、すべてのNPの意味αに対してαが原因となってβが日本語を理解するときにのみ、G（β）（α）＝1となるようなもの（(19d）と(20)、(21a）による）。

c. 【計算機に日本語を理解させる】＝NPの意味αの函数Hで、αが原因となって計算機が日本語を理解するときにのみ、H（α）＝1となるようなもの（(19c）と(21b）による）。

d. 【私は計算機に日本語を理解させる】＝私が原因となって計算機が日本語を理解するときにのみ1となるようなもの（(19b) と (21c) による）。

このようにして文の意味が決まるわけだが、変形文法のやり方と比べると、先に論理学のところで述べたことがそのままあてはまる。すなわち、Frege の原理に基づいた構成的意味論であり、構文的・意味的にまとまった部分の意味に基づいて文全体の意味が決定されるということである。今日、この方法論に基づく意味論の研究は活発に行なわれており、専門家の目から見て不満足なものだったMontague自身の構文論も、文脈自由の句構造文法という観点から検討が行なわれている（例えば Gazdar 1982）。生成文法は長い間変形生成文法であったが、最近になってようやくMontagueの意味論を採用した結果、非変形的生成文法の可能性が生まれてきているのである。

5. 運用論と残された問題

最後に、今まで述べてきた範囲の意味論では扱いきれない問題のいくつかについて簡単に触れておこう。これらは意味論ではなく運用論で扱うべき問題であるが、意味論と密接に関連している。従って、計算機に自然言語を理解させる場合には当然考えなくてはいけないことなのだが、一筋縄ではいかない問題ばかりである。

言外の意味

まず、「言った」ことと「ほのめかした」ことの区別がある（Grice 1975）。これは「論理」と「心理」との違いだと言ってもよいだろう。「言った」ことは論理で扱えるような類いのことであり、「ほのめかした」ことは論理の外に出る。Grice はこの「ほのめかした」ことを扱うために「会話の公準」というものを設定した。これは、「ほのめかした」ことと「言った」こととを切り離し、それぞれが従うべき法則を独立に立ててやろうという考え方である。すなわち、「言った」ことは論理の法則に従っているのに対して、理性をもった人間同士の会話はそれとは別種の公準にも従っているとする。「ほのめかした」ことはこの会話の公準に関連して出てくるのである。こうすることによって、「言った」ことの分析には余計なものを持ちこまずに今までどおりに論理が使えるわけである。

この会話の公準というものは、会話というものは互いに協調的に、相手への「気くばり」をもって行なわれるという大原則に基づいている。この大原則の下に4つの公準がある。第1の「量の公準」は必要にして十分な情報を伝えるということ。舌足らずや喋り過ぎは禁物ということである。第2の「質の公準」は真実のみを伝えるということ。嘘もいけないし、確たる証拠のないことも言ってはいけないということである。第3の公準は内容に関するもので、当然、無関係な話題は避けなくてはいけない。最後の第4の公準は伝え方に関するもので、分りやすくなければいけないということ。

これらの公準は当り前すぎるくらいだが、面白いのは、人間はしばしば意識的にこの公準から外れることがあるということである。そうすることによって、話者は文字通りの意味以外の「言外の意味」をほのめかすことができる。

(22) a. 計算機は言われたことはちゃんとやる。

b. 計算機は気が利かない。

例えば、上の(22a) はその文字通りの意味だけを考えたのでは、誰もが知っていることで情報量はほとんどない。つまり、これは量の公準に違反している。しかし、話者が悪意があって愚にもつかないことを言っているのでない限り、何か「言外の意味」があるはずである。そこで聞き手がたどりつく結論の一つが(22b) ということになる。

ここで注意することは、このような言外の意味のもつ性質の一つに、それが必ずしも一意に決まらないということがある。例えば、(22a) も、言われたことをちゃんとやらない人間を相手に小言を言っている場合では「それに引き換え、おまえは本当に駄目だ」というような言外の意味をもつことになる。こういう点がまさに、気が利かない計算機の中に言外の意味を推察するようなプログラムを組み入れようとする場合には問題となるのである。

前提

言外の意味とは別に、「前提」というものがある。前提とは満たされていないと発話全体が不適切になるようなことである。

(23) 計算機には禿がない。

この文はどことなくおかしい。そのおかしさは、(23)自体が「言っ」ていることよりも、そもそも(23)のような言明が成り立つために前提とされていることが我々の現実世界に対する知識と矛盾しているからである。つまり、Xが計算機であれ、人間であれ、禿があるとかないとかいうことがそもそも言えるためにはその前提としてXに毛があるということが成り立っていなければならない。上の場合、従って前提として計算機に毛がなくてはならないが、これは、第四世代までの計算機を見る限り、事実ではない。次も同じような例である。

(24) 私は人工知能を完成させたことを後悔している。

これも、Xを後悔しているとかいないとかが言えるためには、その前提としてXが真であるということがなければならないということがおかしさの原因となっている。1983年の現在、「私は人工知能を完成させた」と言っても誰も信じてはくれないだろう。

前提は論理的帰結とは区別しなくてはならない。前者は肯定文からも否定文からも同じように出てくることに際立った特徴がある。例えば、(24)の代りに「私は人工知能を完成させたことを後悔していない」と言ってもその前提はやはり「私は人工知能を完成させた」である。こういう性質は論理的帰結にはない。

(25) a. 私は人工知能を完成させたことを証明した。

b. 私は人工知能を完成させたことを証明しなかった。

ある文が前提を持つかどうかは、その文の主動詞による。「後悔する」という動詞は前提を持つが、「証明する」という動詞は前提を持たず、代りに論理的帰結としてその埋め込まれた文が真であることを要求する。従って、(25a) は「私は人工知能を完成させた」という論理的帰結を持つが、論理的帰結である以上、その否定文(25b) から同じ帰結は出てこない。

次は、さらに、上のどちらとも異なった例である。

(26) a. 私は人工知能を完成させたと報告した。

b. 私は人工知能を完成させたと報告しなかった。

「報告する」というような動詞はその埋め込まれた文の真偽には関与しない。嘘の報告ということも可能だからである。従って、(26a) 、(26b) どちらも「私は人工知能を完成させた」とか「私は人工知能を完成させなかった」とかいうことと無関係に言うことができる。

運用論の問題として取り上げられてきている問題は他にも数多いが、今見た二つの問題だけでも一朝一夕には解き難い。言語処理という観点から情報処理機械としての人間と計算機とを比べた場合、前者に対して後者の未熟さを感じざるを得ない。今後の計算機の進歩とそれに応じたソフトウェア技術の進歩が、計算機による自然言語の理解という遠大な理想の実現に寄与するところの多いことを期待してやまない。

文 献

Chomsky, N. (1965), Aspects of the Theory of Syntax, Cambridge, Mass., The MIT Press, 1965.

Gazdar, G. (1982), "Phrase structure grammar," in P. Jacobson and G.K. Pullum (eds.), The Nature of Syntactic Representation, Dordrecht, Holland, D. Reidel, 1982, pp. 131-186.

Gunji, T. (1983), "Generalized phrase structur grammar and Japanese reflexivization," Linguistics and Philosophy, 6, (1983), 115-156.

Grice, H.P. (1975), "Logic and conversation," in D. Davidson and G. Harman (eds.), The Logic of Grammar, Encino, Calif., Dickenson, 1975, pp. 64-75.

Montague, R. (1973), "The proper treatment of quantification in ordinary English," in J. Hintikka, J. Moravcsik, and P. Suppes (eds.), Aproaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics, Dordrecht, Holland, D. Reidel, 1973, pp. 221-242.

Morris, C.W. (1938), "Foundations of the theory of signs," in O. Neurath (ed.), International Encyclopedia of Unified Science, vol. 1, Chicago, University of Chicago Press, 1938, pp. 77-138.